Towards Designing the Best Model for Classification of Fish Species Using Deep Neural Networks

Pranav Thorat, Raajas Tongaonkar and Vandana Jagtap

February 8, 2022

Towards Designing the Best Model for Classification of Fish Species using Deep Neural Networks

Pranav Thorat
Dept. of Computer Engineering
Maharashtra Institute of Technology
Pune, India
thoratpranav@gmail.com

Raajas Tongaonkar
Dept. of Computer Engineering
Maharashtra Institute of Technology
Pune, India
tonraaj@gmail.com

Prof. Vandana S. Jagtap
Dept. of Computer Engineering
Maharashtra Institute of Technology
Pune, India
vandana.jagtap@mitpune.edu.in

***Abstract*—Convolutional Neural Networks have become a powerful tool for classification since 2012. The winners of the ImageNet Challenge have been neural networks for a long time now. Computer vision applications mostly resort to neural networks. This paper extends its application to classify fishes of 23 different species using VGGNet algorithm. The fish images used for training the network is obtained from a live video dataset. We implemented the traditional VGG16 and tried a scaled down version of if too. The results of training these two algorithms were compared on the basis of micro average, macro average and weighted average. VGG16 surpassed VGG8 in almost all parameters but not by a large margin as proved in the results section. Smaller networks consume less memory and take lesser time for training as well as prediction. Hence, smaller networks can be used in simpler applications and can run on a less dedicated hardware setup having restrictions of memory and/or processing power. The reason behind specifically choosing a network with 8 layers is because the number of layers is halved from the well known VGG16 model having a total of 16 layers. Thus, it forms a good comparison problem which has been addressed in the paper below.**

***Index Terms*—Deep Learning, Convolutional Neural Networks, Image Classification, VGGNet.**

I. Introduction

Computer vision is an important topic of research nowadays with an increased amount of work being done in robotics and automation. Underwater image processing introduces additional challenges absent otherwise. Underwater images are essentially characterized by their poor visibility because light is exponentially attenuated as it travels in water and the resulting scenes are poorly contrasted and hazy. The study of the marine ecosystem is important for biological researchers. A subset in marine ecosystems includes the study of underwater living organisms like fishes, crabs, octopuses, whales, seals, etc. Monitoring these organisms in terms of their number and type over time can be useful for the purpose of studying the effect of environmental changes or human actions on these organisms. Hence, if any negative feedback is inferred from such studies, corrective actions can be taken. Therefore the need of a vision based system for the purpose of underwater surveillance is much required. This has led us onto the path of building one such robust system. Here, specifically we are targeting a vision system capable of underwater fish specie detection. These vision systems can be an integral part of underwater vehicles like AUVs (Autonomous Underwater Vehicle) and ROVs (Remotely Operated Vehicles) which can survey large areas of the water bodies non-stop and can provide with useful insights with minimum human intervention. Convolutional Neural Networks have been a useful tool for image detection and classification since 2012. Starting with models like the AlexNet to the recently perceived models like YOLOv3, there have been different CNN algorithms and architectures which have successfully solved the problem of localization and classification with ever increasing classification speed or accuracy or both. In this paper we will describe our approach in the classification of fish species using two different convolutional neural network architectures.The first one is our own Scaled-down VGG16 architecture and the other one is the traditional VGG16 model. Our intention behind using two different models is to compare the accuracy of the models and the time required for training on our dataset. The dataset we are using is called the Fish4Knowledge dataset which belongs to School of Informatics, University of Edinburg, UK. It consists of a total at 27303 images in total belonging to 23 different classes or species. These images have been extracted from a video stream.

II. Related Previous Work

Underwater object detection poses some challenges which are absent otherwise. We have referred to a few research documents which have attempted to work on efficient underwater image processing. Some of them have been described below: Some of the non Machine learning based approaches like imaging sonar have also been used by researchers to detect, track and classify fishes. The performance of the fish detector is evaluated using a set of manually annotated sonar images. The detection results show good agreement with the annotations, although some challenges for future improvement

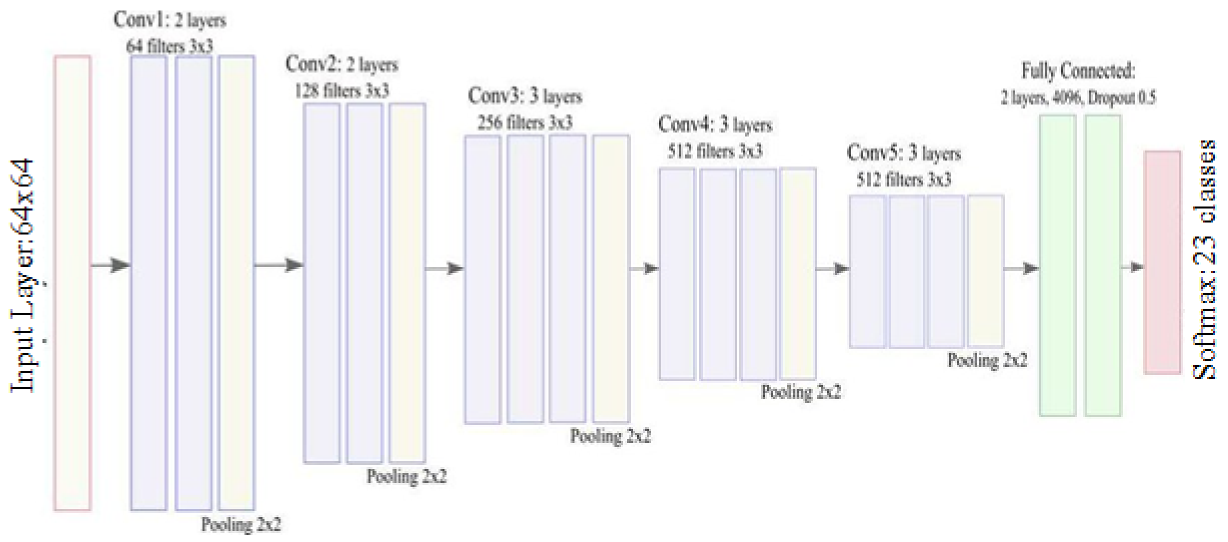

Fig. 1. VGG-16 architecture

of the detector [1] still remain. Some previous research work has been described below:

Many researchers have utilized neural networks to detect fish. Ramani, et al. [2] used parallel networks of three perceptron layers to identify 4 species of fish in sonar images. Storbeck, et al. [3] used a three-layer CNN to devise a way of classifying moving fish. Marburg et al. [4] detected and identified ten classes of benthic macrofauna in optical underwater images using a stack of CNNs. Some researchers have performed detection of underwater objects using neural networks. Byeongjin et al. [5] used AdaBoost algorithm and Haar-like features for object detection in water. Juhwan et al. [6] detected and tracked small ROV in sonar images using a convolutional neural network design. We have also referred to the Google colabatory paper [7] which explains the hardware specification of Colabatory along with its implementation. Also provides a performance review of deploying computer vision, deep learning, classification and other applications on colab.

For object detection many researchers have used very deep convolutional network models like the VGG16 [8], which forms the base of our project, have used networks of increasing depth with small convolutional filters which performed extremely well in the ImageNet Challenge of 2014. The official VGG based models like VGG16 and VGG19 performed the classification task better than its predecessor AlexNet.

Some researchers [9] have performed real-time fish detection on images from actual fish videos using convolutional neural network based techniques based on You Only Look Once algorithm. The network recorded 93% classification accuracy and outperformed older non-CNN based models like classifier trained with histogram of oriented gradient features and support vector machine and sliding window algorithm. The drawback of YOLO classifier is that the accuracy is compromised at the expense of better detection speed.

As mentioned earlier, here we are using two CNN based architectures for classification. Following is the detailed explanation of our network for the fish specie classification.

A. *Previous Architecture*

With reference to [8], we have implemented the traditional VGG16 architecture by Visual Geometry Group at the University of Oxford. While training, the input images to our network are re-sized 64 * 64 RGB images. These images are passed through stacked convolutional layers which are thirteen in number with the number of filters varying in an increasing order in each layer as defined in the original paper and shown in the adjoining figure. The number of filters for each convolutional layer are mentioned in the architecture diagram. Parameter to each convolutional layer includes a small 3 * 3 filter or kernel. Rectified Linear Unit or ReLU is the activation function used. The convolutional stride set to 1 pixel throughout. Five max-pooling layers perform spatial pooling. It is performed over a 2 * 2 pixel window.

The convolutional layers are then followed by a set of three Fully-Connected layers. The first two have 4096 channels each and the third one has 23 channels (one for each class). This final layer is the soft-max layer. Softmax is implemented using a layer just before the output layer. The Softmax layer must have the same number of nodes as the output layer.

III. Proposed Method

In CNN like architecture, selection of parameters like number of layers and arrangement of layer plays an important role in the efficiency, accuracy and performance of our model. Selecting the optimum value of these parameters gives us the best model and helps us avoid tackling problems like model overfitting and excessive time and memory usage during training the neural network. Here we are using a scaled down version of the VGG16 architecture. The similarity between our architecture and the VGG16 architectures is that both of them are uniform and use a small 3 * 3 kernel (or filter) in the convolutional layer. Here the size of the filter (or kernel) is the same (i.e. 3 * 3) throughout the architecture for each convolutional layer which results in reduced number of parameters as compared to a layer of larger using filters of larger size like 7 * 7 (parameters = 47) or 11 * 11 (parameters = 121). Hence, we call it the VGG like architecture.

Also the layers in the architecture are such arranged that the initial layers are the combination of convolutional layers (along with their activation function ReLU) and the pooling layer, in the end, followed by a fully connected layer for class prediction along with its probability value.

With respect to the convolutional layers, our model is partly made up of repetitive stacking of a convolution layer followed by the activation layer and batch normalization. Batch Normalization is used for the purpose of normalizing the activations of the previous layer at each batch. For representational purposes this set is referred to as one unit. Each unit is followed by a pooling layer of window size 2 * 2. After each pooling layer a dropout of 0.25 states the fraction of dropped input unit from the previous layer. After the stack of convolutional layers, next are the two fully connected layer. The first fully connected layer has 512 channels. A dropout of 0.5 in this layer resembles the fraction of input units which are to be dropped. The last layer is the softmax implemented through a fully connected layer just before output layer. The softmax layer has same number of nodes as the output layer, in our case 23 as we dealing with 23 classes.

This algorithm / model we used for the classification of fish species was a scaled down version of the known VGG16 algorithm. The VGG16 architecture has a total of 16 layers but our scaled down VGG16 used a mere 8 layers, 6 convolutional layers and 2 fully connected layer. This particular model can be termed as VGG8. The reason behind using a smaller network is to significantly reduce the time and memory required during training, taking into consideration that the accuracy is not compromised massively as compared to larger convolutional neural network architectures. Also smaller networks are easier and faster to train, hence can be built where hardware resources are a major constraint.

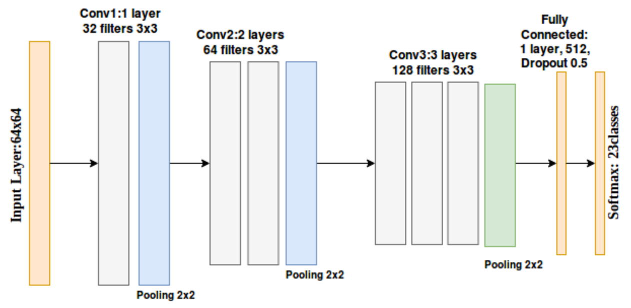

Fig. 2. VGG-8 Architecture

IV. Methodology

Our dataset has been taken from the Fish4Knowledge Project at the University of Edinburgh, School of Informatics, UK. This fish data is acquired from a live video dataset resulting in 27370 verified fish images in total belonging to 23 different fish species.

Following are the steps in which we train our model on the dataset. Initially we load our dataset in the primary memory and resize each image to a 64 * 64 pixel resolution. Next, we split our dataset into training and testing sets, with 75% of the total in training and remaining 25% of the dataset for testing. Next we initialize our CNN model with appropriate parameters and define the hyperparameters and train the network. We have used Google Colaboratory for training purpose. Google colabatory is a cloud service based on Jupyter Notebooks which provides access to a robust GPU. The hardware available is comparable to a mainstream workstation and a robust Linux server equipped with 20 physical cores. For our application, in Google Colabatory, we trained our model on the Python 2 Google Compute Engine Backend (GPU) and a total of 12.72 GB of RAM and 358.27 GB of disk space was allocated.

Two paths were taken -

1) VGG-8: This was basically a smaller version of the VGGnet algorithm. It had 6 convolutional layers and 2 fully connected layer
2) VGG-16 : This was the implementation of the actual VGG-16 algorithm. It has 13 convolutional layers and 3 fully connected layers.

In both these implementations, the number of epochs was the same, 75. We kept the batch size 32 with an initial learning rate of 0.01 The activation function for both implementations was Rectified Linear Unit (ReLU), with the final fully connected layer having Softmax activation function.

Epochs	Batch Size	Learning Rate
75	32	0.01

TABLE I
Hyperparameters

V. Results

It can be observed from both the graphs that as the iterations during training increases there is a gradual decrease in the loss parameters i.e. the training loss and validation loss values. On the other hand, accuracy parameters like the training accuracy and the validation accuracy increase gradually as the number of epochs increase. We have also tried to choose the optimum value of the hyperparameters such that the model does not overfit.

VGG16 had a higher initial training loss than VGG8. Loss value implies how poorly or how well a certain model behaves after each iteration of optimization. Training loss is the average

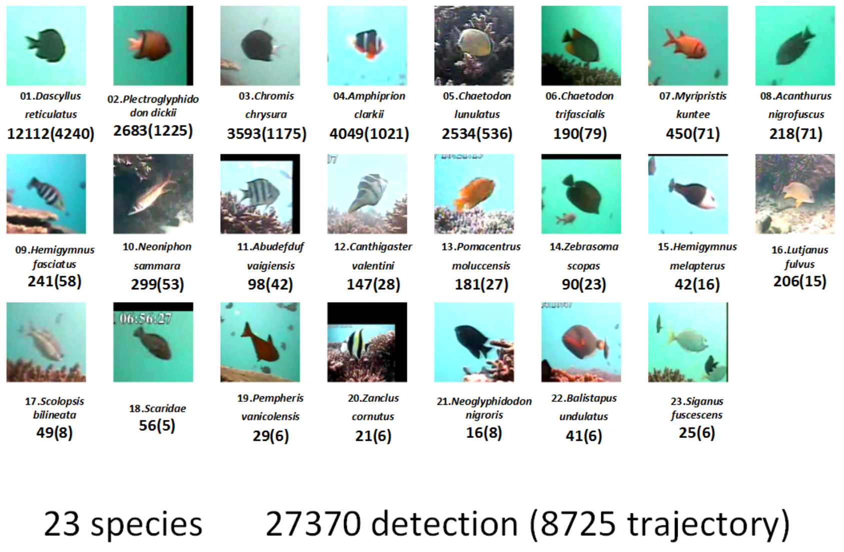
01.Dascyllus reticulatus 12112(4240)
02.Plectroglyphidodon dickii 2683(1225)
03.Chromis chrysura 3593(1175)
04.Amphiprion clarkii 4049(1021)
05.Chaetodon lunulatus 2534(536)
06.Chaetodon trifascialis 190(79)
07.Myripristis kuntee 450(71)
08.Acanthurus nigrofuscus 218(71)
09.Hemigymnus fasciatus 241(58)
10.Neoniphon sammara 299(53)
11.Abudefduf vaigiensis 98(42)
12.Canthigaster valentini 147(28)
13.Pomacentrus moluccensis 181(27)
14.Zebrasoma scopas 90(23)
15.Hemigymnus melapterus 42(16)
16.Lutjanus fulvus 206(15)
17.Scolopsis bilineata 49(8)
18.Scaridae 56(5)
19.Pempheris vanicolensis 29(6)
20.Zanclus cornutus 21(6)
21.Neoglyphidodon nigroris 16(8)
22.Balistapus undulatus 41(6)
23.Siganus fuscescens 25(6)
23 species
27370 detection (8725 trajectory)

Fig. 3. Dataset

of the losses over each batch of training data. Testing or validation loss is computed using the model as it is at the end of an epoch. It was almost equal to training loss for VGG16, and lower than training loss for VGG8. Both achieved a training accuracy and validation accuracy greater than 0.95 by the 50th epoch. Overall, VGG-16 performed better than VGG-8. However, the increase in performance was marginal. VGG-8 is thus a better option in situations where there is a restriction on the resources available for training the model. The table shows the average value for the given metric for the 23 classes. Macro average computes the metric independently for each class and then takes the average. Hence, all classes are treated equally. Micro average on the other hand will aggregate the contributions of all classes to compute the average metric. In weighted average, instead of each data point contributing equally to the average, some data points contribute more than others. VGG-8 has a marginally higher macro average value of the performance metric of precision. VGG-16 scores higher in all other metrics. VGG-16 is thus only marginally better than VGG-8.

	VGG-16	VGG-8
Micro average	Precision: 0.99 Recall: 0.99 F1-score: 0.99	Precision: 0.98 Recall: 0.98 F1-score: 0.99
Macro average	Precision: 0.94 Recall: 0.89 F1-score: 0.90	Precision: 0.96 Recall: 0.86 F1-score: 0.90
Weighted average	Precision: 0.99 Recall: 0.99 F1-score: 0.98	Precision: 0.98 Recall: 0.98 F1-score: 0.98

TABLE II
PERFORMANCE METRICS

For testing purpose, following is a random image of the fish specie Amphiprion Clarkii downloaded from Google Images and tested on our model, which it correctly classifies and

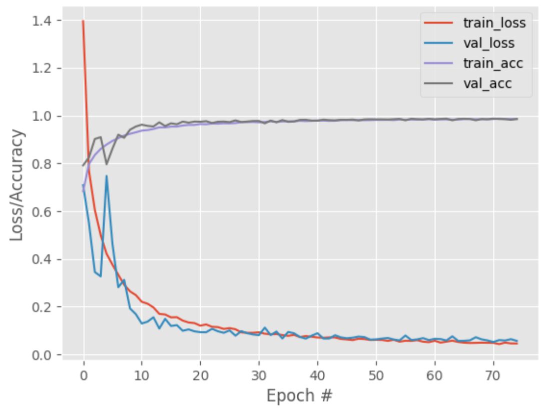

Fig. 4. VGG-16

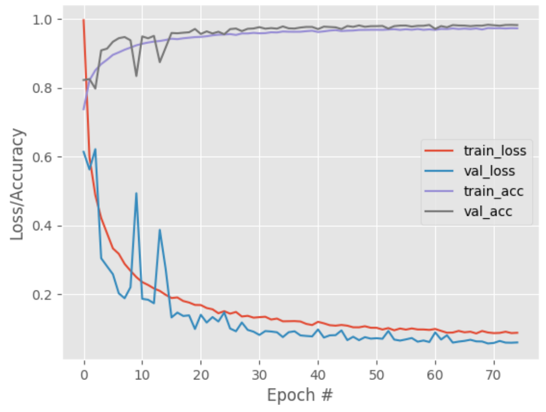

Fig. 5. VGG-8

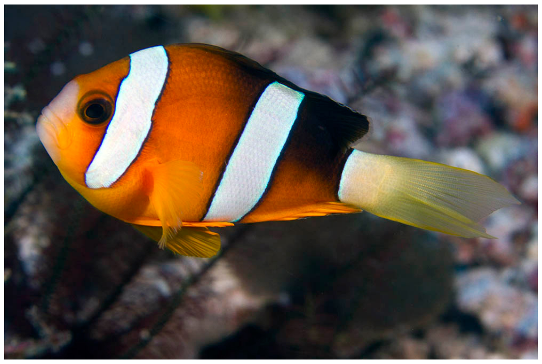

Fig. 6. Amphiprion Clarkii (Source:Google Images)

Species	Precision	Recall	f1-score	Support
Abudefduf vaigiensis	1.00	0.91	0.95	22
Acanthurus nigrofuscus	0.88	0.67	0.76	64
Amphiprion clarkii	1.00	1.00	1.00	1007
Balistapus undulatus	1.00	0.70	0.82	10
Canthigaster valentini	0.87	0.98	0.92	41
Chaetodon lunulatus	1.00	1.00	1.00	618
Chaetodon trifascialis	0.98	0.98	0.97	46
Chromis chrysura	0.99	0.99	0.99	900
Dascyllus reticulatus	0.98	0.99	0.99	3039
Hemigymnus fasciatus	0.95	1.00	0.97	57
Hemigymnus melapterus	1.00	0.33	0.50	12
Lutjanus fulvus	1.00	1.00	1.00	52
Myripristis kuntee	0.99	0.99	0.99	105
Neoglyphidodon nigroris	1.00	0.50	0.67	4
Neoniphon sammara	1.00	0.99	0.99	74
Pempheris vanicolensis	1.00	1.00	1.00	6
Plectroglyphidodon dickii	0.99	0.99	0.99	677
Pomacentrus moluccensis	1.00	1.00	1.00	45
Scaridae	0.88	1.00	0.93	14
Scolopsis bilineata	0.89	1.00	0.95	16
Siganus fuscescens	0.67	0.86	0.75	4
Zanclus cornutus	0.67	0.86	0.75	7
Zebrasoma scopas	0.87	0.57	0.68	23

TABLE III
VGG16 RESULTS

gives a significant accuracy of 98.25% on the VGG8 model. Whereas, the same image when tested using the VGG16 classifier correctly classified the specie but with a lesser accuracy of 96.07%. One of the possibility that might explain this result is that a larger network might have lead to overfitting. Hence, we can can conclude that a larger network might not always lead to better results as observed in this case. Thus, our proposed architecture is better suited for this dataset than the standard VGG16 model. Following is a comparative study of the two models.

VI. Conclusion and Future Scope

In this paper, a CNN based approaches for fish classification were implemented and analysed. Specifically, two models, VGG16 and VGG8 were trained on our dataset. Both models showed similar performance in terms of various performance measures. The differentiating factor between both the models is the time required for training and the amount of space used up. The VGG8 model is better than VGG16 considering time

Species	Precision	Recall	f1-score	Support
Abudefduf vaigiensis	0.91	0.91	0.91	22
Acanthurus nigrofuscus	0.94	0.52	0.67	64
Amphiprion clarkii	1.00	1.00	1.00	1007
Balistapus undulatus	1.00	0.70	0.82	10
Canthigaster valentini	0.88	0.93	0.90	41
Chaetodon lunulatus	0.99	1.00	1.00	618
Chaetodon trifascialis	1.00	0.87	0.93	46
Chromis chrysura	0.98	0.99	0.98	900
Dascyllus reticulatus	0.98	0.99	0.99	3039
Hemigymnus fasciatus	0.97	1.00	0.98	57
Hemigymnus melapterus	1.00	0.67	0.80	12
Lutjanus fulvus	1.00	1.00	1.00	52
Myripristis kuntee	1.00	0.98	0.99	105
Neoglyphidodon nigroris	1.00	0.50	0.67	4
Neoniphon sammara	0.99	0.99	0.99	74
Pempheris vanicolensis	1.00	1.00	1.00	6
Plectroglyphidodon dickii	0.99	0.99	0.99	677
Pomacentrus moluccensis	1.00	1.00	1.00	45
Scaridae	1.00	0.86	0.92	14
Scolopsis bilineata	0.78	0.88	0.82	16
Siganus fuscescens	1.00	1.00	1.00	4
Zanclus cornutus	0.83	0.71	0.77	7
Zebrasoma scopas	0.82	0.39	0.53	23

TABLE IV
VGG8 RESULTS

and memory utilization. Hence, models like the VGG8 can be used when there is a restriction in the hardware resources available with the user. Also, such lighter models can be used in simple classification problems in which not much detailing in the images is required for classification, whereas, denser models like VGG16 can be used in much problems requiring much precise detailing/analysis of images for classification. We have also tested our model on images which are outside of the dataset and the results are pretty satisfactory and accurate which is the first step in building a robust system. Hence, the above model can be deployed on an AUV or ROV wherein as the underwater vehicle moves around it will pick up images and if fishes present will be correctly classified and these results can be used for analysing the water body by the biological researchers. We have recently built our own AUV in collaboration with another team and will soon deploy our classification module on the AUV.

In future, the above model can be tested more rigorously. The system can be trained to work efficiently in more challenging background environments and also in low lighting conditions. Also, this model can be improved to be built on a more uniformly distributed images in the dataset with respect to each class.

References

[1] L. M. Wolff, S. Badri-Hoeher ”Imaging sonar-based fish detection in shallow waters” 2014 Oceans - St. John’s Please number citations consecutively within brackets

[2] Ramani, Narayan, and Paul H. Patrick. ”Fish detection and identification using neural networks-some laboratory results.” IEEE journal of oceanic engineering 17.4 (1992): 364-368.

[3] Storbeck, Frank, and Berent Daan. ”Fish species recognition using computer vision and a neural network.” Fisheries Research 51.1 (2001): 11-15.

[4] Kim, Byeongjin, and Son-Cheol Yu. ”Imaging sonar based real-time underwater object detection utilizing AdaBoost method.” Underwater Technology (UT), 2017 IEEE. IEEE, 2017.

[5] Kim, Juhwan, and Son-Cheol Yu. ”Convolutional neural network-based real-time ROV detection using forward-looking sonar image.” Autonomous Underwater Vehicles (AUV), 2016 IEEE/OES. IEEE, 2016.

[6] Marburg, Aaron, and Katie Bigham. ”Deep learning for benthic fauna identification.” OCEANS 2016 MTS/IEEE Monterey. IEEE, 2016.

[7] Tiago Carneiro, Raul V. Medeiros Da Nbrega, Thiago Nepomuceno, Gui-bin Bian, Victor Hugo C. De Albuquerque , and Pedro P. Rebouas Filho ”Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications” IEEE Access (Accepted but unpublished).

[8] Karen Simonyan Andrew Zisserman. ”Very Deep Convolutional Neural Networks for Large Scale Image Recognition” ICLR 2015.

[9] Minsung Sung and Son-Cheol Yu Yogesh Girdhar. ”Vision based Real-time Fish Detection Using Convolutional Neural Network” OCEANS 2017 - Aberdeen.

[10] ”Convolutional Neural Network” Wikipedia - The Free Encylopedia